MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2015/2016

ECE3136 – JAVA TECHNOLOGY

(All Sections / Groups)

17 OCTOBER 2015
9:00 A.M – 11:00 A.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This examination paper consists of SIX (6) pages including cover with THREE (3) questions only.

2. Answer ALL questions. All questions carry 25 marks and the distribution of the marks for each question is given.

3. This is an Open-Book examination. Materials permitted are textbooks and bound notes.

4. Please print all your answers in the Answer Booklet provided.

Question 1

Write a program that checks the number of characters entered by the user to ensure that the number of characters does not exceed the given limit. Your program should contain the following classes:

(a) Create a class **`StringChecker`** which has an instance variable of type **`String`**, `input`. Provide a constructor.

[4 marks]

(i) Create a method **`checkLength`** which takes in a **`String`**, **`input`** and an **`int`**, **`limit`**. Check that the length of the string is less than the limit. If it exceeds return true, else return false.

[4 marks]

(ii) Create a method **`hashString`** which takes in a **`String`**. Hash the string and return the hash integer.
(Hint: use **`String`** class method **`hashCode()`** which returns an **`int`**. See Figure Q1.)

[2 marks]

`int`	`hashCode()` Returns a hash code for this string.

Figure Q1: Java API reference for String class hashcode method

(b) Create the main driver class **`ProcessStrings`**, which has as instance variables: an **`int`** constant called **`MAX`**, a **`String`**, **`input`** and a **`StringChecker`**, **`checker`**.

[2 marks]

(i) Read a string from the user. Call the method **`checkLength`** from the **`StringChecker`** class to check the number of characters. If the number of characters in the string is less than the limit (30 characters), then print the string. Else, if the string entered has too many characters, throw an exception. This exception is called the **`StringTooLongException`**), designed to be thrown when a string is discovered to have too many characters in it.

[5 marks]

(ii) Catch and handle the exception by printing an appropriate message and continue processing more strings from the user.

[6 marks]

(iii) If the user's string is within the limit, then call the method **`hashString`** to output the hash of the string.

[1 mark]

(iv) After user has entered all desired strings, user ends interaction by entering the phrase "DONE" as the string.

[1 mark]

Continued...

Question 2

Create a class **`BMICalculator`** which calculates Body Mass Index (BMI) based on input given by the user.

$$\text{BMI} = \frac{Weight(kg)}{(Height(m))^2}$$

These are the BMI Categories:

Underweight = BMI<18.5
Normal weight = BMI of 18.5 to 24.9
Overweight = BMI of 25 to 29.9
Obese = BMI of 30 or greater

Your program should contain the following:

(a) The class **`BMICalculator`** inherits from **`JPanel`** and has three labels, two text fields, three subpanels and a slider. The three subpanels are for displaying height, weight with slider and BMI as shown in Figure Q2. Instantiate the height text field to have a default value of 1.6 and the weight text field to have a default value of 40. Set the weight text field to be non-editable, as the weight will be input by dragging the slider.

[8 marks]

(b) Set up the slider object to have a value ranging between 40 to 170, inclusive, with a default value of 40. Attach a **`ChangeListener`** to the slider object. Add the subpanels to the content pane.

[7 marks]

(c) Provide a method, **`setRating`**, which changes the text at the bottom panel according to the BMI. These are the rating levels: Underweight, Normal weight, Overweight and Obese. It takes as an argument the current BMI value.

[4 marks]

(d) Inside the listener, write the **`stateChanged`** method which updates the display every time the slider is dragged.
The displayed BMI value is to have one decimal point.
To change the rating in the bottom panel, call method **`setRating`** and pass it the current value of the weight slider.

[6 marks]

Note: DO NOT write the driver program.

Continued...

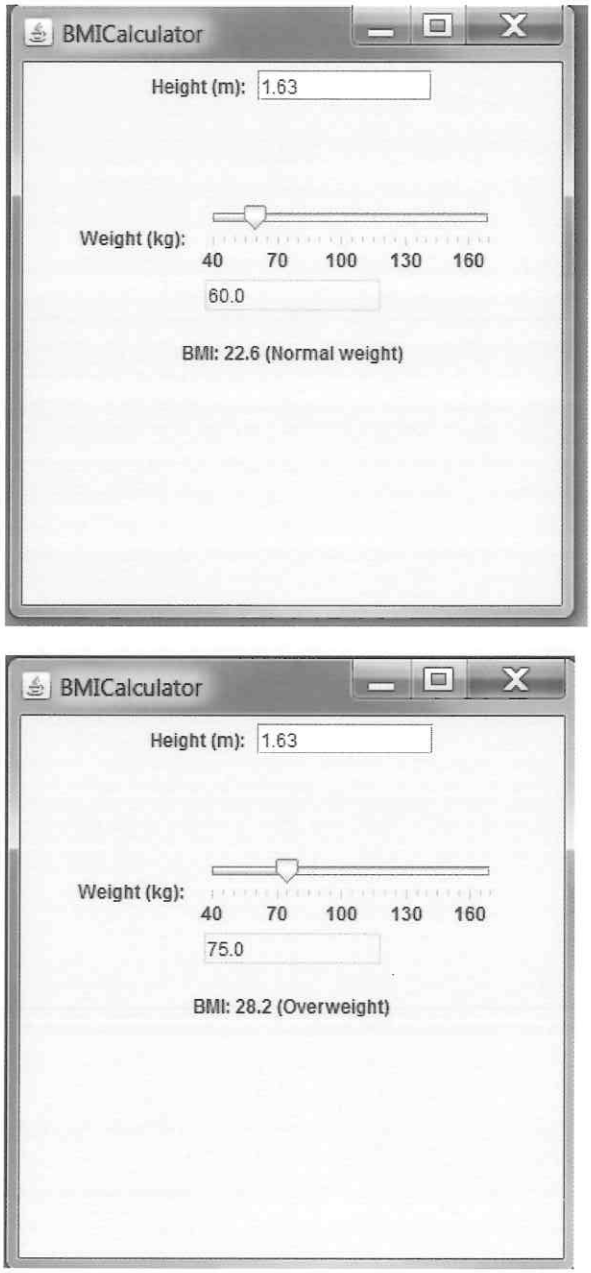

Figure Q2: BMI values change when the slider is dragged

Continued...

Question 3

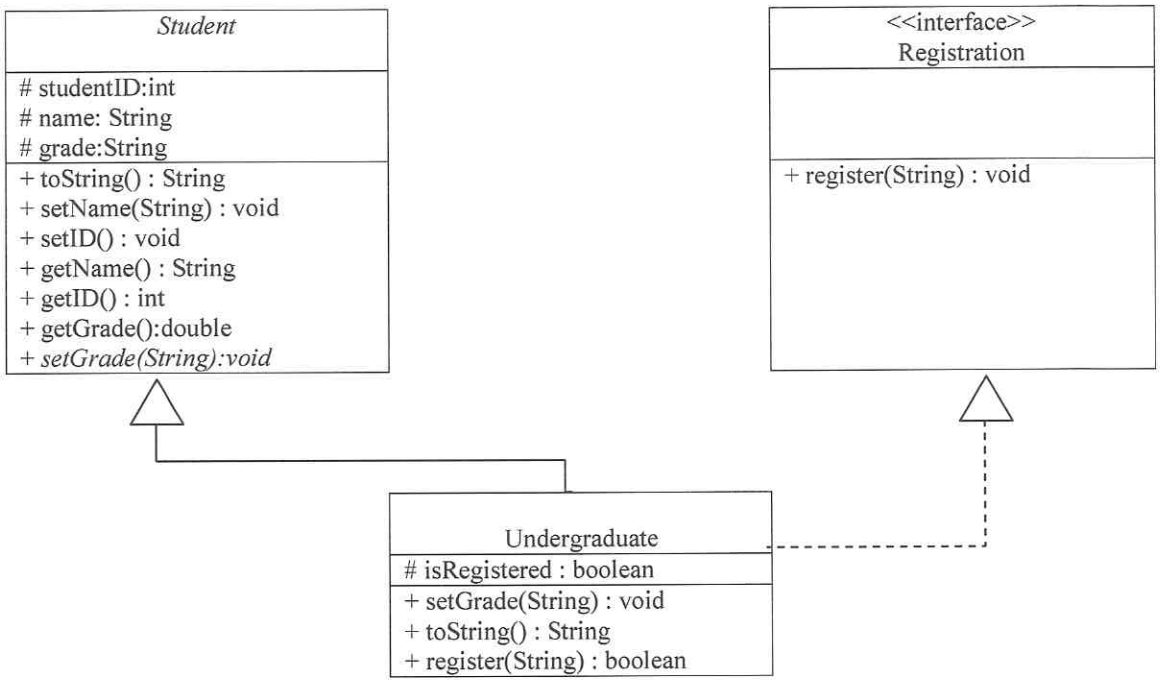

Figure Q3

Refer to the UML diagram in Figure Q3. In your programs, include all class members shown in the UML diagram.

(a) Create abstract class **`Student`**.

(i) Provide mutator and accessor methods. The mutator must only allow the value of **`studentID`** to be set to a number ranging from 10000000 to 99999999. Output an error message otherwise. Provide a constructor, a **`toString`** method and the abstract method **`setGrade`**.

[11 marks]

(b) Create interface **`Registration`**.

[2 marks]

(c)

(i) Create a class **`Undergraduate`** which inherits from **`Student`** and implements interface **`Registration`**. It has a variable of type **`boolean`**, called **`isRegistered`**. Override the superclass **`toString`** method.

[3 marks]

(ii) Provide a constructor which uses the superclass constructor. It takes in the arguments **`name`**, **`studentID`**, a CGPA **`grade`** eg. 3.92 and a **`previousDegree`** eg. "Foundation".

[2 marks]

Continued...

(iii) Override abstract method **`setGrade`**. The method **`setGrade`** should only allow the grade to be set if the value is between 0 and 4. Output an error message otherwise.

[2 marks]

(iv) Override abstract method **`register`**. The method **`register`** should check the String input on whether it matches (irrespective of case) either "FOUNDATION" or "FORM 6". If yes, **`isRegistered`** is true, else it is false.

[2 marks]

(d) Create a simple driver class to test **`Undergraduate`** class. The driver program must print out all the instance variables' values. It should also include cases with erroneous arguments to check error detection.

[3 marks]

End of Paper